VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Aplikace pro přihlašování uživatelů do systému Linux pomocí RFID karet

Applications for Logging Users Into the Linux System with RFID Cards

2014 František Fiala

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **František Fiala**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Aplikace pro přihlašování uživatelů do systému Linux pomocí RFID karet
Applications for Logging Users Into the Linux System with RFID Cards

Zásady pro vypracování:

Úkolem bakalářské práce je vytvoření aplikace, která bude schopna číst informace z RFID karty a pomocí těchto informací se bude možné přihlásit do OS Linux bez toho, aby bylo nutné zadávat přihlašovací údaje. Aplikace by měla být navržena pro pracovní prostředí GNOME a měla by být integrována mezi ostatní přihlašovací mechanizmy. Předpoklad je, že finální aplikace bude spolupracovat se systémem PAM nebo bude implementována přímo do přihlašovacího manažeru GDM, popř. kombinace těchto možností.

1. Popište zařízení pro čtení RFID karet.
2. Popište standardní přihlašovací mechanizmy v OS Linux především v prostředí GNOME.
3. Navrhněte a pokuste se realizovat program, který umožní přihlášení pomocí RFID karty v prostředí GNOME.
4. Při programování dbejte všech dobrých zásad pro psaní programů v OS Linux.
5. Svou práci dobře zdokumentujte a popište způsob implementace systému.

Seznam doporučené odborné literatury:

[1] kolektiv autorů. Linux - Dokumentační projekt. 4. vydání. Brno: Computer Press, 2007. ISBN: 978-80-251-1525-1
[2] Geisshirt, Kenneth. Pluggable Authentication Modules: The Definitive Guide to PAM for Linux SysAdmins and C Developers. 1. vydání. Birmingham: Packt Publishing Ltd., 2007. ISBN 978-1-904811-32-9

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Seidl, Ph.D.**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 7. 5. 2014 ..... Fiala

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. 5. 2014 ..... Fiala

Na tomto mieste by som rád poďakoval pánovi Ing. Davidovi Seidlovi, PhD. za odbornú a ochotnú pomoc ako aj za jeho rady, ktoré mi v práci veľmi pomohli.

Abstrakt

Cieľom tejto práce je navrhnút'a implementovat'systém, ktorý umožní užívateľovi prihlásenie do systému Linux pomocou RFID karty. Práca popisuje prihlásovacie a autentifikačné techniky. Využíva pri tom PAM framework. Na čítanie kariet bolo použité zariadenie ID - 12 Innovations. Výsledok práce bol testovaný na distribúcii Debian 7.

Klíčová slova: autentifikačné moduly, pam, prihlásenie, autentifikácia, ldap, linux, rfid

Abstract

The goal of this thesis is to design and implement a system that lets the user to log in using a RFID card. Thesis describes logging and authentification mechanisms. It takes advantage of PAM framework. As a card reading device ID - 12 Innovations was used. Results were tested on distribution Debian 7.

Keywords: authentification modules, pam, log in, authentification, ldap, linux, rfid

Seznam použitých zkratek a symbolů

RFID	–	Radio-frequency identification
PAM	–	Pluggable authentification modules
API	–	Application programming interface
LDAP	–	Lightweight directory access protocol
MG	–	Management group

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Obsahom mojej práce je navrhnúť aplikáciu, ktorá umožní prihlásenie do systému Linux pomocou technológie RFID bez nutnosti zadávania užívateľského mena a hesla.

Pri práci budeme využívať čítacie zariadenie ID - 12 Innovations spolu s pasívnymi RFID tagmi, ktorých držiteľom je každý študent a zamestnanec VŠB-TU Ostrava. Aplikáciu budem programovať v jazyku C++. Ako systém, pre ktorý sa budem snažiť dosiahnúť vyššie uvedeného cieľa som zvolil linuxovú distribúciu Debian 7 s kódovym označením Wheezy.

2 Technológia RFID

2.1 Popis technológie

RFID je skratka od **Radio-Frequency IDentification**, čo v preklade znamená identifikácia pomocou rádiovej frekvencie. V podstate ide o malé elektronické zariadenie, ktoré pozostáva z čítacieho zariadenia a malého čipu. Tento čip je schopný uchovať približne 2 KB dát ale vo väčsine prípadov je to viac ako dosť.

RFID zariadenie slúži k podobnému účelu ako napríklad čiarové kódy alebo magnetické prúžky. Poskytuje unikátnu identifikáciu nejakému objektu. Tak ako pri čiarových kódoch, aj v tomto prípade platí, že informácia slúžiaca k identifikácii objektu sa najskôr musí naskenovať. Ako skenovacie zariadenie slúzi RFID anténa. Informáciu nesie RFID tag, čo môže byt karta, známka ale existujú aj miniatúrne tagy, ktoré sa používaju napr. k identifikácii zvierat. Informácia o identite vo forme unikátnej sekvencie znakov sa prenáša bezkontaktne prostredníctvom rádiových vĺn.

2.2 Ako to funguje

Technológia RFID pozostáva z troch častí:

- Skenovacia anténa
- Prijímač spolu s dekóderom (transceiver)
- Transpondér (RFID tag) s naprogramovanou informáciou

2.2.1 RFID anténa

RFID anténa generuje rádiové signály, ktoré majú relatívne krátky dosah. Toto pole má za úlohu dve veci:

- Slúži ako kominikačný kanál na prenos informácii
- Poskytuje RFID tagu energiu aby sa mohol uskutočnit' prenos informácie (toto platí iba pri pasívnom tagu)

Práve táto druhá vlastnosť je kľúcová. RFID tagy nemusia niesť zdroj energie a preto sa dajú používať po veľmi dlhú dobu.

Keď sa RFID tag dostane do poľa, ktoré anténa generuje, detekuje od nej aktivačný signál. Tento signál "prebudí"tag a následne dochádza k prenosu informácie, ktorú obsahuje.

2.2.2 RFID tag

Tagy slúžia ako vlastné nosiče informácie o identifikácii objektu. Tagy rozdeľujeme do dvoch základných skupín:

- Aktívne tagy
- Pasívne tagy

Aktívne tagy nesú vlastný zdroj energie. Výhoda týchto tagov spočíva v tom, že vzdialenosť čítacieho zariadenia a tagu môže byť väčšia. Aj keď sú tagy navrhuté tak, aby vydržali aktívne po dobu približne 10 rokov, ich životnosť je obmedzená. Toto je hlavná nevýhoda tejto skupiny tagov.

Pasívne tagy nevyžadujú zdroj energie. Môžu byť menšie a majú prakticky neobmedzenú životnosť. K poruche väčšinou dochádza mechanickým poškodením alebo vystavením veľmi silnému elektromagnetickému poľu.

RFID tagy je možné použiť v širokej škále prípadov, v ktorých sú technológie ako napríklad čiarové kódy alebo magnetické prúžky nepoužiteľné.

- Tag nemusí byť umiestnený na povrchu objektu
- Čas čítania je zvyčajne menší ako 100 ms
- Je možné čítať viac tagov v jednom okamihu

2.2.3 Prijímač a dekóder

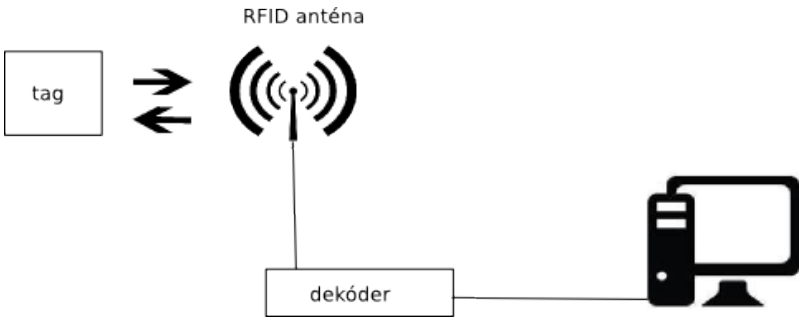
Prijímač a dekóder spolu s anténou tvoria čítacie zariadenie. Anténa dostane od tagu informácie vo forme elektrických signálov a následne je potrebné tieto signály spracovať do podoby, ktorá je použiteľná v počítači. Táto časť zariadenia taktiež zabezpečuje prenos informácie do počítača. Informácie sa prenášaju podľa komunikačného protokolu, ktorý sa pri rôznych typoch zariadení môže odlišovať.

2.3 Problémy s technológiou RFID

Žiadna technológia nie je úplne dokonalá, pri každej vznikajú určité problémy, ktoré treba brať do úvahy. Výnimkou nie je ani technológia RFID. V následujúcich pár riadkoch sú vymenované niektoré základne.

2.3.1 Technické problémy

2.3.1.1 Problém s RFID štandardom Systémy využívajúce RFID sú v rôznych prípadoch implementované rôzne. Do teraz neexistuje žiadny globálny štandard, ktorý by umožnoval bezproblémovú interakciu naprieč rôznymi systémami. Treba však spomenúť, že táto technológia na niečo takéto nebola navrhnutá.



Obrázek 1: Schéma RFID zariadenia

2.3.1.2 Systém môže byť ľahko narušený Vhľadom na to, že informácie sa prenášajú prostredníctvom rádiových vĺn, systém môže byť ľahko "zablokovaný"generovaním elegtromagnetického poľa so správnou frekvenciou v okolí antény.

2.3.1.3 Kolízia čítacích zariadení Tento problém nastáva v prípade, kedy sa polia generované dvoma alebo viacerými čítacími zariadeniami prekrývajú. RFID tag nie je schopný komunikovať s viacerými zariadeniami v jednom okamihu. Systém musí byť navrhnutý tak, aby sa takýmto kolíziam predišlo.

2.3.1.4 Kolízia tagov Nastáva v prípade, kedy sú viaceré tagy veľmi blízko pri sebe.

2.3.2 Bezpečnosť, súkromie a etické problémy

2.3.2.1 RFID tag môže byť prečítaný bez Vášho vedomia Nakoľko proces čítania nevyžaduje priamy kontakt bez prekážky (tak ako je to pri čiarovom kóde alebo magnetickom prúžku), každý, kto ma k tomu potrebné vybavenie je schopný prečítať informáciu uloženú na tagu. Následne je možné tag zduplikovať a tým ukradnúť identitu.

2.3.2.2 RFID tag môže byť prečítaný na väčšiu vzdialenosť s použitím vysoko výkonnej antény Aj keď je sýstém obvykle navrhnutý tak, že pri čítaní musí byť vzdialenosť medzi tagom a anténou minimálna, s použitím antény s veľkým výkonom je možné čítať aj na väčšiu vzdialenosť. Tento fakt taktiež vedie k možnému narušeniu súkromia alebo odcudzeniu identity.

3 Zariadenie ID - 12 Innovations

Zariadenie tvorí modul, ktorý obsahuje prijímač spolu so vstavanou anténou. K počítači sa pripája cez USB port. Tým zabezpečime zariadeniu aj potriebné napájanie 5V. Technologické parametre zariadenia su znázornené v tabuľke 1. Schéma zapojenia modulu je na obrázku 2 Po pripojení sa v systéme tvári ako sériový port RS232. Tento port nájdeme v adresári. */dev/ttyUSB0*.

3.1 Formát dátového výstupu

Po tom, ako vložíme RFID kartu do poľa antény, dostaneme od zariadenia sekcenciu 16 ASCII znakov:

- STX - predstavuje začiatok prenosu dát, hexadecimálna hodnota 0x02
- 10 znakov predstavujúcich dáta o identite
- 2 znaky predstavujúce kontrolný súčet
- znak CR
- znak LF
- ETX - predstavuje koniec prenosu dát, hexadecimálna hodnota 0x03

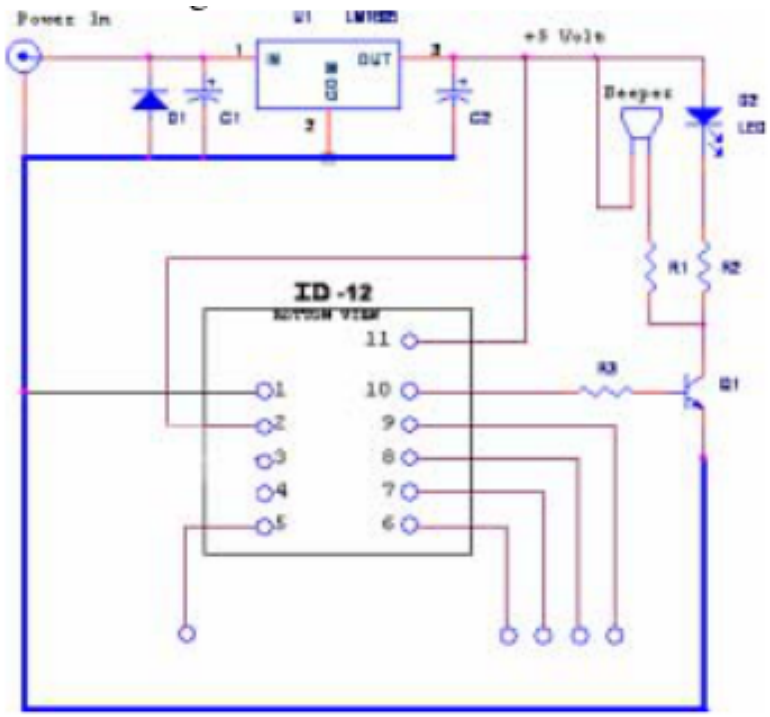
Pre naše potreby nás bude zaujímat iba časť s dátami o identite.

3.2 Čítanie dát v jazyku C++

Pre zostavenie jednoduchého programu v jazyku C++ budeme potrebovať hlavičkové súbory *termios.h, sys/ioctl.h*. Ako bolo spomenuté vyššie, čítacie zariadenie sa v systéme javí ako obyčajná sériová linka. Každá sériova linka prenáša dáta určitou rýchlosťou (baudrate). Na tento fakt nesmieme pri programovaní zabúdať. Naše zariadenie využíva rýchlosť 9600. Preto je potrébné po otvorení portu zmenit toto nastavenie viď príklad 1. Následne môžme začat čítať dáta s použitím funkcie *read()*.

```
int port;
port = open(parameters.device, O_RDONLY);
if (port == -1)
{
    printf ("Cannot open card reading device.\n");
   return 1;
}
/* PORT OPTIONS */
struct termios opt;
tcgetattr (port, &opt);
cfsetispeed(&opt, B9600);
cfsetospeed(&opt, B9600);
tcsetattr (port, TCSANOW, &opt);
```

Výpis 1: Otvorenie portu na čítanie



Obrázek 2: Schéma zapojenia

Dosah pri čítaní	12+ cm
Rozmery	26 mm x 25 mm x 7 mm
Frekvencia	125 kHz
Formát karty	EM 4001 alebo kompatibilné
Kódovanie	Manchester 64-bit, modulus 64
Napájanie	5 VDC, 30 mA
Rozsah napájacieho zdroja	+4.6 V až 5.4V

Tabulka 1: Parametre zariadenia

4 Prihlásenie v systéme linux

Predtým, ako začneme pracovať na počítači, je nutné prejsť procesom prihlásenia. Výnimku netvorí ani systém linux. Keďže linux tak ako ostatné operačné systémy podporuje prístup viacerých užívateľov, proces prihlásenia je nutný z dôvodu oddelenia a ochrany dát. Tento proces pozostáva z dvoch častí

- Autentifikácia
- Autorizácia

Je veľmi dôležité nemýliť si tieto dva pojmy. **Autentifikácia** je proces, pri ktorom sa zisťuje identita užívateľa. Zjednodušene povedané zisťuje sa, či užívateľ je naozaj ten, za koho sa vydáva. **Autorizácia** je proces, pri ktorom sa zisťujú práva užívateľa na prácu s dátami. Napr. či má daný užívateľ v určitý čas k dátam prístup.

4.1 Prihlásenie v režime tty

Po tom, ako systém nabootuje, užívateľ vidí v termináli nápis *menostroja login:*. Tento nápis je generovaný procesom **getty**. Tento proces je znova spustený procesom **init** po tom, ako sa užívateľ odhási alebo je neúspešný proces prihlásenia. Program getty volá program login a ten volá shell v prípade, že proces prihlásenia je úspešný. Kroky tohoto procesu v poradí

1. Proces init spustí proces getty.
2. Proces getty si vyžiadá užívateľské meno. Po tom ako ho užívateľ zadá, spustí program login a užívateľské meno mu predá ako argument.
3. Program login si vyžiadá heslo a skontroluje. Ak je heslo správne, spustí shell inak sa program ukončí a proces init znova spustí proces getty.
4. Užívateľ je prihlásený. Pri odhlásení sa program ukončí a vrátime sa ku kroku číslo 1.

4.1.1 Proces init

Proces init má pid 1 a je rodičovským procesom pre všetky ostatné. V súbore */etc/inittab* sa náchadza riadok *1:2345:respawn:/sbin/getty tty1*. Tento riadok spôsobí, že v runleveloch 2,3,4,5 bude vždy bežať proces getty a v prípade, že by sa ukončil, proces init ho spustí znova. Toto docielí tak, že použije funkciu "fork"na vytvorenie svojej kópie a následne použije funkciu "exec"pre spustenie programu getty.

4.1.2 Program getty

Getty má za úlohu nasledujúce funkcie

1. Otvorenie tty linky a nastavenie jej módu
2. Získať užívateľské meno
3. Spustiť proces prihlásenia užívateľa

Po tom, ako sa nastaví "terminálová linka"program getty pošle na výstup obsah súboru */etc/issue*. Následne prečíta užívateľské meno a invokuje program login, ktorému predá toto meno ako argument. Počas čítania užívateľského mena sa getty pokusí adaptovať systém na rýchlosť použitého terminálu.

Tty zariadenie, ktoré getty používa sa určuje parametrom. Parametre sú definované v súbore */etc/inittab*.

4.1.3 Program login

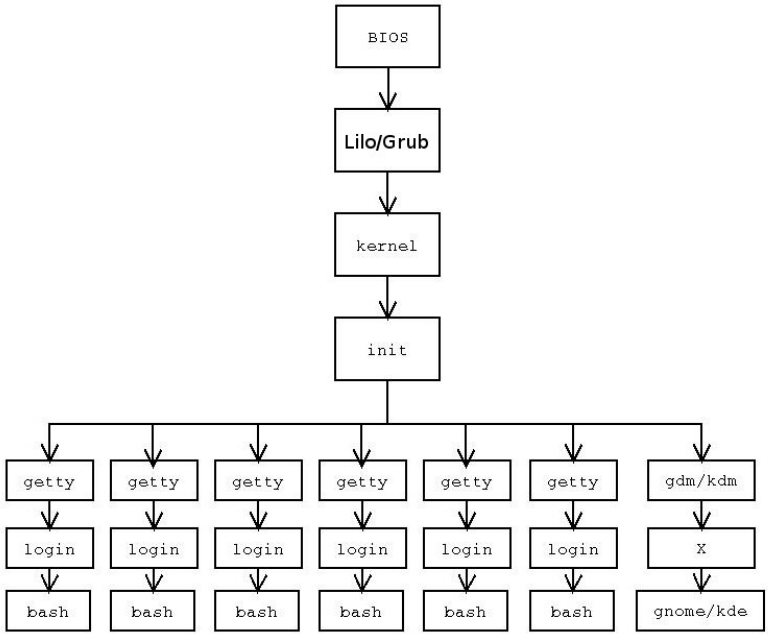
Ak je program login spustený bez argumentu, požiadá si o uživateľské meno. Ak existuje súbor */etc/nologin* a užívateľ nie je root, vypíše obsah tohto súboru na výstup. Ak sú v súbore */etc/usrtty* nastavené nejaké špeciálne obmedzenia prístupu, skontrolujú sa a v prípade chyby bude prihlásenie prerušené a syslog zaznamená pokus o prihlásenie. Ak je uživateľ root, prihlásenie musí byť z terminálu, ktorý je zahrnutý v súbore */etc/securetty*.

Záznamy o užívateľských účtoch sa za normálnych okolností nahrávajú zo súboru */etc/passwd*. Záleží to však na nastavení Name Service Switch v konfiguračnom súbore */etc/nsswitch.conf*. Okrem lokálne uložených záznamov je možné nahrávať účty napríklad z ldapu alebo databázy.

Príklad záznamu o užívateľskom účte

username:password:UID,GID,GECOS:homedir:shell

- username - Užívateľské meno
- password - Šifrované heslo alebo zástupny znak v prípade, keď systém uchováva hesla v súbore */etc/shadow*, ktorý môže čítať iba root
- UID - Unikátne číslo pridelené k účtu
- GID - Číslo pridelené k hlavnej skupine užívateľa
- GECOS - Zvyčajne celé meno užívateľa. Slúži iba pre informáciu a je nepovinné
- homedir - Cesta k domovskému priečinku
- shell - Určuje, ktorá konzola sa spustí po úspešnom prihlásení



Obrázek 3: Diagram procesu zavedenia systému

Následuje proces autentifikácie. Najskôr sa overí, či užívateľ v systéme existuje. Ak áno, spustia sa autentifikačné mechanizmy. Autentifikáciu zabezpečujú PAM moduly a sú bližšie opísané v samostatnej sekcii. V prípade, že autentifikácia prebehne úspešne, proces prihlásenia je dokončený a užívateľ môže začať systém používať.

4.2 Prihlásenie v grafickom prostredí

Prihlásenie do grafického prostredia zabezpečujú iné aplikácie, ale proces autentifikácie je praktický totožný s tým v režime tty. Aj tu sa pre autentifikáciu používajú PAM moduly.

Nakoľko v tomto prípade nepotrebujeme žiadnú terminálovú linku, nespúšťa sa program getty ale program gdm. Program gdm je akousi alternatívou programu login v grafickom prostredí X. Gdm vykreslí vstupné pole na zadanie užívateľského mena spolu s tlačídlami pre login a zrušenie prihlásenia. Následne sa zavolajú PAM moduly a ak autentifikácia prejde, spustí sa X server a následne grafické prostredie, ktoré ma daný užívateľ nastavené.

5 PAM moduly

PAM je skratka od Pluggable Authentification Modules, čo v preklade znamená zásuvné autentifikačné moduly. PAM dokážu mnoho vecí avšak primárne zameranie je autentifikácia užívateľov. Okrem toho dokáže ešte napríklad nastaviť prostredie, v ktorom užívateľ pracuje. Keď sa užívateľ odhlási, PAM sa dokážu postarať aj o uvoľnenie zdrojov, ktoré súvisia s daným prostredím.

5.1 História PAM

História PAM siahá späť do roku 1995, kedy vývojarský tým z Sun Microsystems implementoval generický framework pre Solaris. Keď bol Auguste roku 1997 vydaný Solaris 2.6, PAM sa stali jeho integrovanou komponentou. Od vtedy Solaris používa PAM ako primárny systém pre autentifikáciu užívateľov. Vo Februári roku 1997 začal projekt Linux-PAM a v dnešnej dobe ho využíva väčsina GNU/Linux distibúcií.

Primárny operáčný systém tejto práce je GNU/Linux, ale PAM projekt existuje pre mnoho operačných systémov. Naprieč operáčnými systémami typu UNIX sú konfiguračné súbory takmer identické. Môžu sa odlišovať iba mena modulov a niektoré moduly nemusia byť všade podporované.

5.2 PAM rieši problém autentifikácie

Ako už bolo spomenuté skôr, pred tým ako užívateľ začne pracovať na počítači, musí určitým spôsobom preukázať svoju identitu - absolvovať proces autentifikácie a autorizácie. Vo väčšine prípadov musí poskytnúť užívateľské meno a heslo. Každá aplikácia, ktorá vyžaduje autentifikáciu, musi implementovať vlastné autentifikačné mechanizmy. Problém sa prejaví, keď pridáme viaceré aplikácie vykonávajúce ten istý druh autentifikácie. Prihásenie do grafického prostredia vyžaduje display manager, ktorý musí byť schopný validovať užívateľov. Teraz pridajme služby ako FTP, TELNET, IMAP, SSH a ďalšie aplikácie, ktoré vyžadujú určitý druh autentifikácie. Ako systémovy administrátor by ste trávili veľa času udržovaním a spravovaním mnoho databáz užívateľov popri */etc/passwd*. Spravovanie takejto množiny databáz užívateľov by sa mohlo stať nočnou morou v prípade, že by sa vyskytla v databázach nejaká nekonzistencia. Naviac každý užívateľ by si musel pamätať meno a heslo pre každú z týchto aplikácii.

5.3 Potreba PAM

PAM a aplikácie, ktoré využívajú PAM redukujú zložitosť autentifikácie. S PAM je možné využívať jednu databázu užívateľov pre všetky aplikácie ktoré to vyžadujú. Naviac je možné použiť viac spôsobov autentifikácie kontrolovanej PAM a pre užívateľa to ostáva transparentné. Výhodou je tiež to, že znalosť PAM na jendom operačnom systéme môže byť ľahko prenesená na iný.

PAM majú dobre definované API a aplikácie ktoré to využívajú nespadnú v prípade, že systémovy administrátor zmení konfiguráciu autentifikácie.

Naviac, súbor */etc/passwd* sa nedá škálovať. Funguje dobre pri 100 užívateľoch, ale ak pracujeme s 5000 užívateľmi, určite pocítime rozdiel. S PAM je veľmi jednoduché v prípade potreby zmeniť databázu užívateľov z lokálneho súboru napríklad na LDAP server. Toto by bez PAM mohol byť dosť veľký problém.

Programovanie aplikácie ktorá vyžaduje autentifikáciu taktiež prináša výhody ak použijeme PAM. Nie je nutné implementovať mnoho funkcií, ktoré autentifikáciu zabezpečia. Výber databázy užívateľov ako aj použité autentifikačné mechanizmy nechávame na výber systémovemu administrátorovi.

5.4 PAM Famework

PAM ako generický framework využíva dynamicky načítavané moduly (implementované ako dynamické knižnice - súbory *.so). Modul môže poskytovať mechanizmus na autentifikáciu užívateľa na základe informácie z rôznych zdrojov.

Každý modul je vlastne jedna dynamická knižnica a môže poskytovať 4 základne druhy bezpečnostných služieb

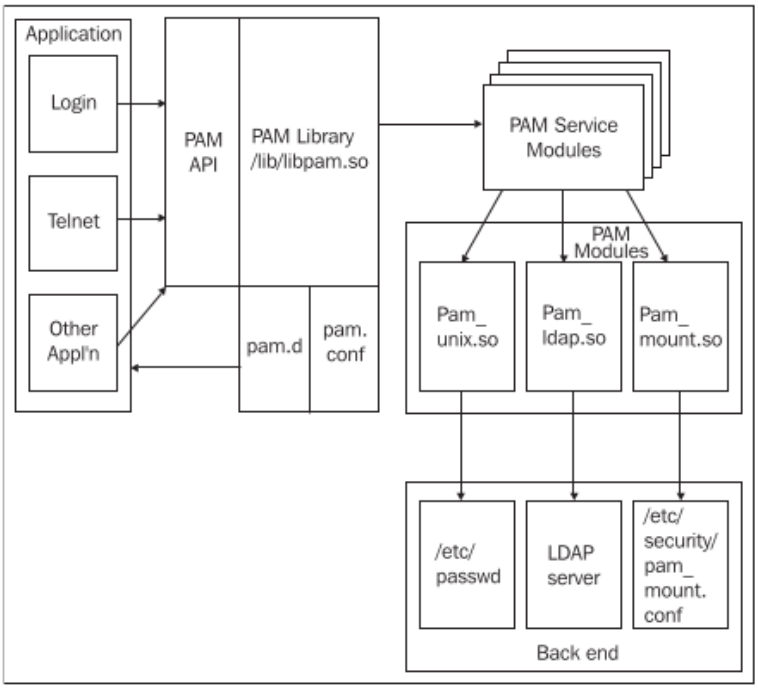
- Služby pre autentifikáciu (AUTH)
- Služby pre správu účtov (ACCOUNT)
- Služby pre správu relácii (SESSION)
- Služby pre správu hesiel (PASSWORD)

Každý modul môže implementovať jednu alebo viac týchto služieb. Aplikácia, ktorá využíva PAM komunikuje s modulom prostredníctvom PAM API. Moduly poskytujú funkcie potrebné pre autentifikáciu, správu účtov, nastavenie prostredia a správu hesiel.

Konfiguráciou v PAM frameworku je možné zaistiť viaceré techniký validácie užívateľa počas jedného pokusu prihlásenia. Môžme vyžadovať, že všetky moduly musia skončiť úspechom aby bolo prihlásenie úspešné, alebo môžme vybrať jeden modul, ktorý na autentifikáciu postačuje. Túto autentifikačnú politiku je možné nastavovať podľa potreby a to bez toho, aby sme museli prekompilovať aplikáciu, reštartovať systém alebo plánovať odstávku.

```
auth   required pam_unix.so nullok_secure
auth   optional pam_mount.so use_first_pass debug
auth   optional pam_ssh.so use_first_pass debug
```

Výpis 2: Príklad konfigurácie



Obrázek 4: Architektúra PAM frameworku

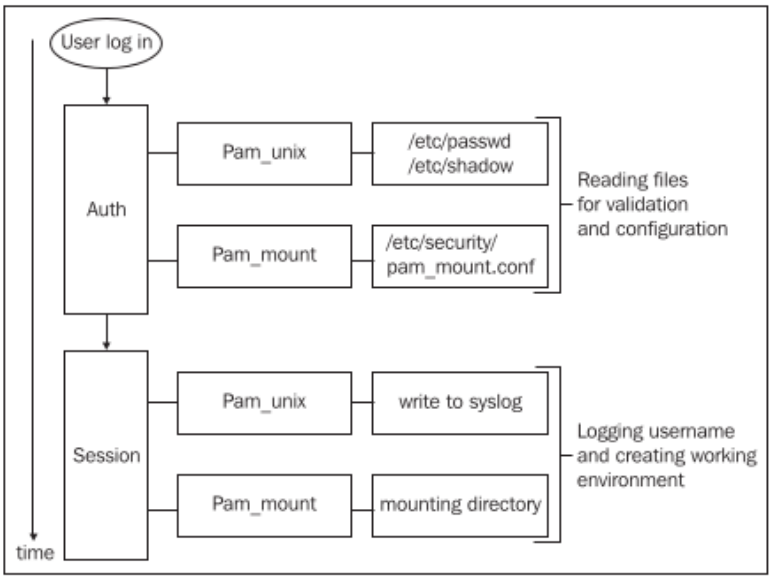
5.5 Služby

Aplikácie, ktoré vyžadujú autentifikáciu sa u PAM môžu registrovať pomocou názvu PAM služby. Názov služby je daný aplikáciou a určuje sa pri inicializácii PAM frameworku volaním funkcie *pam_start*. Politika autentifikácie je definovaná v konfiguračných súboroch v priečinku */etc/pam.d*. Platí, že názov služby je totožný s názvom konfiguračného súboru. V prípade, že sa konfiguračný súbor s daným názvom v adresári nenachádza, použije sa súbor *other*. Ak neexistuje adresár */etc/pam.d*, politika autentifikácie je definovaná v jedinom súbore */etc/pam.conf*.

5.6 Management Groups

Každá PAM služba môže využívať moduly v 4 štádiach autentifikácie. Tieto štádia sa v PAM nazývajú Management Groups. Každý modul môže poskytovať potrebnú funkcionalitu pre jedno alebo viac týchto štádií. Časový priebeh pri prihlásení užívateľa je znázornený na obrázku 5.

- MG AUTH - Postkytuje 2 funkcie. Prvá *pam_authenticate* zabezpečuje sammotnú autentifikáciu. Druhá *pam_setcred* nastavuje alebo upravuje informácie o užívateľovi v závislosti od kontextu.
- MG ACCOUNT - Poskytuje funkciu *pam_acct_mgmt*, ktorá predstavuje autorizáciu.



Obrázek 5: Priebeh prihlásenia

- MG SESSION - Poskytuje 2 funkcie. Prvá *pam_open_session* pripravuje prostredie po autentifikácii. Druhá *pam_close_session* uvoľňuje zdroje, ktoré súvisia s prostredím.
- MG PASSWORD - Poskytuje funkciu *pam_chauthtok*, ktorá zabezpečuje zmenu hesla. V našom prípade by to bola zmena RFID karty.

5.7 Konfigurácia politiky

Ako bolo spomenuté skôr, autentifikačné mechanizmy aplikácie sa nastavujú v konfugiračných súboroch v adresári */etc/pam.d*. Užitočnou vlastnosťou PAM frameworku je "stackovanie"modulov. Pre každú MG je možné definovať zoznam modulov, ktoré budú zabezpečovať mechanizmy súvisiace s daným štádiom autentifikačného procesu.

Keď aplikácia zavolá nejakú funkciu z PAM knižnice (napríklad *authenticate*), PAM služba definovaná aplikáciou pri inicializácii PAM frameworku načíta daný konfiguračný súbor a vykoná danú funkciu z každého modulu. Následne na základe politiky vyhodnotí výsledok a aplikácii vráti informáciu o tom, či autentifikácia prebehla úspešne. Poradie vykonávania funkcií z modulov závisí na poradí modulov definovaných v konfiguračnom súbore. Na toto treba dávať veľký pozor, pretože na poradí záleží a zmena poradia jednotlivých modulov môže mať veľký dopad na spôsob autentifikácie.

V konfiguračnom súbore sa zádáva každý modul na vlastný riadok. Vyžadujú sa uviesť 3 informácie.

- Management group

- Kontrolný flag
- Názov modulu (súboru .so)

Ďalej je možné zadať argumenty daného modulu avšak táto položka je nepovinná. Konfiguráciu je možné rozdeliť do viacerých súborov a následne vkladať direktívou *@include*. Forma jedného riadku konfigurácie je uvedená nižšie.

management group kontrolný flag názov modulu argumenty

5.8 Kontrolné flagy

Každý modul môže skončiť buď úspechom alebo neúspechom. Výsledky jednotlivých modulov môžu mať pre nás rôznu váhu. Niektoré môžu byť dôležitejšie ako iné a preto je nutný nejaký spôsob, ako to vyjadriť pri konfigurácii. Práve na toto slúžia kontrolné flagy, ktoré určujú priebeh autentifikácie a majú veľký vplyv na konečný výsledok o úspechu alebo neúspechu autentifikácie. Kontrolné flagy sa zadávajú do druhého stĺpca konfigurácie a rozlišujeme 4 základne druhy.

- REQUISITE
- REQUIRED
- SUFFICIENT
- OPTIONAL

V operačnom systéme Solaris od verzie 8 a viac existujú ešte niektoré ďalšie.

5.8.1 REQUISITE

Tento flag je pravdepodobne ten najsilnejší. Ak je modul označený ako REQUISITE a skonči chybou, okamžite sa zastaví priebeh vykonávania autentifikácie a aplikácia dostane správu o neúspechu. Všetky moduly uvedené za týmto budú v tomto prípade ignorované.

5.8.2 REQUIRED

Návratova hodnota z modulu označeného ako REQUIRED sa uloží. V prípade chyby vykonávanie nie je prerušené a pokračuje sa ďalším modulom. Keď sa vykoná celý stack modulov a aspoň jeden modul označený flagom REQUIRED skončí chybou, aplikácia dostane správu o neúspechu. Aj keď je chyba autentifikácie asociovaná s prvým REQUIRED modulom ktorý vráti chybu, na strane aplikácie nie je možné zistiť kde sa stala chyba. Toto môže byť pre prípadneho útočníka nevýhoda nakoľko nevie, ktorou fázou autentifikácie neprešiel.

5.8.3 SUFFICIENT

Tento flag môže byt'v niektorých prípadoch celkom silný. Vykonávanie stacku sa zastaví, ak modul označený týmto flagom vráti OK. Ak sa pred ním nenáchadzal žiádný REQUIRED modul, ktorý skončil chybou, aplikácia okamžite dostáva správu o úspechu a ďalsie REQUIRED moduly budú ignorované. Flag je istým spôsobom opak flagu REQUISITE.

5.8.4 OPTIONAL

Ak je modul označený týmto flagom, vôbec neovplyvňuje vykonávanie stacku tak ako iné moduly. Naviac návratova hodnota nemá žiadny vplyv na konečný výsledok. Takéto moduly zvyčajne slúžia na to, aby vykonali nejaké dopĺnkové činnosti súvisejúce s autentifikáciou s tým, že nie sú povinné.

6 Vývoj s PAM frameworkom

V tejto kapitole si povieme, ako vyvýjat' aplikácie využívajúce PAM framevork pre autentifikáciu. Aj keď existuje mnoho modulov, ktoré riešia mnohé autentifikačné techniky a mechanizmy, môže nastat' situácia, kedy potrebujeme naprogramovat' vlastný modul. V tejto časti si ukážeme ako.

6.1 Vývoj aplikácie využívajúcej PAM

Základná myšlienka PAM frameworku je taká, že samotná aplikácia nemusí "vediet"nič o tom, akým spôsobom sa užívateľ autentifikuje. Pre takúto aplikáciu je postačujúce, aby dostala správu o tom, či má alebo nemá poskytnút' danému užívateľovi svoju funkcionalitu. PAM framework to rieši tak, že aplikácia si nalinkuje PAM runtime library a následne len volá potrebné funkcie z PAM API. O samotnú autentifikáciu sa starajú moduly. Na to aby sme mohli pracovat's PAM API, je potrebné vlozit' hlavičkový subor *pam_appl.h*.

```
#include <security/pam_appl.h>
#include <security/pam_misc.h>


static struct pam_conv conv = {
misc_conv,
NULL
};


int main()
{
    int retval;
    pam_handle_t *pamh = NULL;

    retval = pam_start("myapp", "user", &conv, &pamh);
    if (PAM_SUCCESS == pam_authenticate(pamh,0))
    {
        // auth success
    }
    else
    {
        // auth error
    }

}
```

Výpis 3: Príklad jednoduchej aplikácie ktorá využíva PAM

Pri inicializácii PAM frameworku zadávame 4 parametre. Prvý je názov PAM služby, ktorá sa postará o autentifikáciu. Taktiež určuje názov konfiguračného súboru. Druhý

je užívateľské meno. Tretí je referencia na štruktúru, ktorá nesie meno konverzačnej funkcie. V prípade linuxu nie je potrebné ju implementovať a nachádza sa v hlavičkovom súbore *pam_misc.h*. Posledným parametrom je referencia na štruktúru, ktorá predstavuje PAM handler. Vidíme, že aplikácia nerieši autentifikačné mechanizmy. Tie by museli byť definované v konfiguračnom súbore */etc/pam.d/myapp*

6.2 Konverzačná funkcia

Konverzačná funkcia zabezpečuje komunikáciu medzi modulom a aplikáciou. Využíva sa napríklad pri vypísovaní chýb, informácii ale aj získavaní hodnôt od užívateľa. Toto je potrebné kvôli tomu, že PAM môžu využívať rôzne aplikácie či už konzolové alebo aj grafické. Tu je rozdiel v interakcií s užívateľom zrejmý. Ako už bolo spomenuté skôr, v linuxe je táto funkcia už implementovaná v hlavičkovom súbore *pam_misc.h*.

6.3 Vývoj PAM modulu

Vývoj vlastného PAM modulu spočíva v implementovaní funkcií z PAM API. Sú to funkcie, ktoré sú asociované s jednotlivými štádiami autentifikácie. Rozdiel v názvoch funkcií spočíva pridaní *sm* do názvu funkcie. To znamená, že napríklad funkciu *pam_authenticate* pri implementácii modulu pomenujeme *pam_sm_authenticate*. Taktiež musíme pridať hlavičkový súbor *pam_modules.h*.

Každá funkcia asociovaná s určitým štádiom autentifikácie má množinu návratovych kódov, ktoré môže vrátiť. Na základe týchto kódov potom PAM služba podľa definovanej politiky určuje konečný výsledok, ktorý aplikácia dostane. Zoznam týchto kódov spolu s významami a štádiami autentifikácie, ktorých funkcie ich môžu vrátiť je znázornený v tabuľke 2.

Návratový kód	Štadia autentifikácie	Význam
PAM_SUCCESS	všetky	Všetko prebiehlo v poriadku
PAM_USER_UNKNOWN	auth,account,password	Neznámy užívateľ
PAM_SESSION_ERR	session	Problém s reláciou
PAM_AUTH_ERR	auth,account	Chyba pri autentifikácii
PAM_ACCT_EXPIRED	account	Účet expiroval

Tabulka 2: Návratové hodnoty funkcií

Po implementovaní jednej alebo viacerých funkcií sa program zkompiluje a výsledná dynamická knižnica sa presunie do adresára */lib/security*. Týmto je modul pripravený a môže sa začať používať. Príklad súboru make, ktorý slúži na kompiláciu a nainštalovanie modulu je uvedený vo výpise kódu 4.

```
all :
    g++ -fPIC -c -lpam -lldap -lcrypto *.cpp
    g++ -shared -o pam_rfid_mapping.so *.o -lpam -lldap -lcrypto
clean:
    rm *.o
    rm *.so
install :
    cp pam_rfid_mapping.so /lib/security/
```

Výpis 4: Príklad make súboru pre PAM modul

7 Návrh riešenia

Táto časť práce bude bližie popisovať to, ako s využitím predošlých znalostí navrhnúť a využiť v praxi systém aplikácii a modulov, ktoré umožnia prihlásenie užívateľa s pomocou RFID karty. Popisuje radu problémov, ktoré pri tom vznikajú, ako nakonfigurovať systém a modifikácie existujúcich aplikácii.

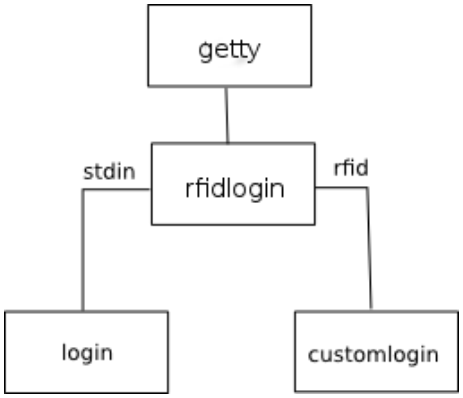
7.1 Čo treba naprogramovať

- Aplikáciu, ktorá sleduje 2 vstupy súčastne a na základe vstupu invokuje buď to program login alebo program customlogin. Táto aplikácia bude slúžiť pre prihlásenie v režime tty.
- Upraviť program login tak aby bolo možné oddelene nastavovať autentifikačné mechanizmy pre prihlásenie RFID kartou. Týmto vznikne program customlogin, ktorý sa bude dať konfigurovať v súbore */etc/pam.d/customlogin*. Týmto programom sa budeme prihlásovať v prípade ak užívateľ zvolí prihlásenie kartou. Ako užívateľské meno budeme očakávat dáta z karty.
- PAM modul pre program customlogin, ktorý bude mapovať užívateľa na základe dát z karty - pam_ldap_mapping.
- PAM modul pre program gdm, ktorý zaistí to, že ak zadáme predom zvoleného užívateľa, modul načíta dáta z karty a následne zmení užívateľa na základe identifikácie. Tento modul bude fungovať aj pre všetky ostatné aplikácie, vyžaduje však, aby sa zadal predom zvolený užívateľ. V prípade, že to bude iný užívateľ, modul vráti chybu a autentifikácia pokračuje ako za normálnych okolností. Tento modul sa bude volať pam_rfid_mapping.

7.2 Prihlásenie v režime tty

Ako už vieme z predošlých sekcií, proces init spúšta program getty. Program getty ešte pred tým, ako spustí program login, vyžaduje od užívateľa meno. Tomuto musíme zabrániť tým, že konfiguráciu v súbore */etc/inittab* zmeníme tak, aby sa program getty spustil s parametrom *-n*. Týmto zabránime tomu aby program getty čakal na užívateľské meno a miesto toho ihneď spustil program login.

Program getty umožnuje parametrom *-l* definovať, ktorý program bude spúšťať. Za normálnych okolností sa spúšťa program */bin/login*. My ale potrebujeme okrem štandardného vstupu sledovať ešte vstup z čítacieho zariadenia na RFID karty. Preto do konfigurácie pridáme ešte parameter *-l rfidlogin*, čím dosiaheme, že program getty spustí rfidlogin. Ten bude sledovať štandardný vstup a vstup z čítacieho zariadenia. V prípade, že dáta prídu zo štandardného vstupu, program rfidlogin spustí program login a vstup mu predá ako argument. Tým padom sa vykoná štandardné prihlásenie ako za normálnych okolností. V prípade, že dáta prídu z čítacieho zariadenia, prečíta sa obsah RFID tagu a spustí sa



Obrázek 6: Diagram priebehu prihásenia po úprave

upravená verzia programu login - customlogin s tým, že dáta z RFID tagu sa mu predajú ako argument. Úprava spočíva v tom, že pri inicializácii PAM sa zmení názov PAM služby a tým aj konfiguračný súbor, ktorý definuje autentifikačné mechanizmy. Vlastne teda dosiahneme to, že pre každý vstup máme vlastný kofiguračný súbor a tým pádom je možné zvlášť nastaviť autentifikáciu pre vstup z klávesnice a zvlášť pre vstup z čítacieho zariadenia.

```
1:2345:respawn:/sbin/getty -l /bin/ rfidlogin -n 38400 tty1
```

Výpis 5: Výsledná konfigurácia getty v /etc/inittab

Konfugurácia autentifikácie pre login ostáva nezmenená teda bude fungovať tak, ako pred tým. Do konfigurácie pre program customlogin je potrebné pridať modul pam_ldap_mapping.

Grafický znázornený priebeh pri pihlasovaní je vidieť na obrázku 6

7.3 Prihlásenie do grafického prostredia

Program, ktorý zabezpečuje prihlásenie do grafického prostredia je program gdm. V tomto prípade použijeme jedného predom zvoleného užívateľa na to, aby sme vedeli, kedy sa chce uživateľ autentifikovať kartou. Takže je potrebné do systému pridať jedného užívateľa - napríklad *rfid*. Na tento účet sa nikdy nebude možné prihlásiť. Slúži iba na to, aby pam modul zdetekoval, že má prečítať obsah RFID karty. PAM modul následne na základe získanej informácie vyhľadá užívateľa v LDAPe a ak existuje, zmeni uid.

```
#%PAM-1.0
auth       requisite            pam_nologin.so
auth   required pam_succeed_if.so user != root quiet_success
auth    sufficient    pam_succeed_if.so user ingroup nopasswdlogin
```

```
@include common-auth
auth    optional        pam_gnome_keyring.so
```

Výpis 6: Konfigurácia autentifikácie programu gdm

Aby sme užívateľovi umožnili okamžité prihlásenie pomocou RFID karty, v súbore */etc/gdm3/daemon.conf* povolíme autologin a nastavíme na predom zvoleného užívateľa. V našom prípade je to užívateľ *rfid*. Týmto zabezpečime to, že pri štarte počítača bude automatický vyžadovaná autentifikácia pomocou karty. Ak by sa chcel užívateľ prihlásit' zadaním údajov, môže proces prihlásenia prerušit' tlačidlom a normálne zadat' svoje prihlásovacie údaje.

7.4 Modul pam_rfid_mapping.so

Modul slúži na to, aby zmenil užívateľa na základe informácie z RFID karty. Túto informáciu si prečíta priamo zo zariadenia, ktoré definujeme parametrom *device*.

- Parameter user - Týmto parametrom sa definuje, ktorého užívateľa bude modul používat' na to, aby načítal dáta z čítacieho zariadenia a na základe nej vyhľadal užívateľa v LDAPe.
- Parameter device - Cesta k súboru predstavujúcemu čítacie zariadenie. V nášom prípade je to /dev/ttyUSB0.

Modul poskytuje iba funkciu authenticate. Ako prvý krok, modul získa meno užívateľa, ktorý sa chce prihlásit'. Na toto sa použije funkcia *pam_get_item*. Ak sa užívateľské meno zhodje s parametrom user, je to pre modul signál, že bolo zadané užívateľské meno slúžiace na mapovanie na základe dát z RFID karty.

Tento modul je navrhnutý tak, aby sa označil kontrolným flagom SUFFICIENT nakoľko ak sa užívateľské meno nezhoduje s tým v argumente, modul vráti chybu. Takto by sa v prípade označenia flagom napr. REQUIRED nedalo prihlásit' na žiadného iného užívateľa okrem toho v parametri. Toto riešenie som zvolil kvôli tomu, že máme v systéme jedného užívateľa napríklad *rfid* a tento slúži na to, aby modul načital dáta z karty a na základe nich namapoval správneho. V inom prípade modul vráti chybu a keďže je označený flagom SUFFICIENT, v autentifikácii sa pokračuje ďalej a tento modul sa ignoruje.

V prípade, že sa uživateľské meno zhoduje s tým v argumente, otvorí zariadenie definované v argumente device a čaká na vstup z RFID karty. Ak prídu dáta, modul sa pripojí na LDAP server a na základe nich výhľadá užívateľa. Ak užívateľ v LDAPe existuje, pomocou funkcie *pam_set_item* nastaví užívateľské meno na meno získane z LDAPu. Následne vráti správu o úspechu autentifikácie. Nakoľko v konfigurácii autentifikácie je nastavený s kontrolným flagom SUFFICIENT, v autentifikácii sa ďalej nepokračuje a aplikácia dostáva správu o úspechu spolu s novo nastaveným užívateľom.

Môžu teda nastať dve situácie:

- Pokus o prihlásenie užívateľa, ktorý je definovaný v konfigurácii PAM modulu.
- Pokus o prihlásenie iného užívateľa.

7.5 Modul pam_ldap_mapping.so

Tento modul využijeme pri konfigurácii autentifikácie programu customlogin. Je praktický totožný s predošlým modulom. Slúži k tomu, aby mapoval užívateľov na základe dát, ktoré na rozdiel od modulu pam_rfid_mapping.so dostane ako užívateľské meno.

```
PAM_EXTERN int pam_sm_authenticate(pam_handle_t *pamh, int flags, int argc, const char *argv
    [])
{
   int ret;
   LDAP                    *ld;
   LDAPMessage             *result;

    const void* hash;
    ret = pam_get_item(pamh,PAM_USER,&hash);


    ret = lc_connect(&ld,"ldaps://ldap.vsb.cz", LDAP_VERSION3, "cn=tuo_card_id_reader,ou=
        home,o=cvt", "nenacteno");
   if (ret != 0)
   {
      pam_info(pamh, "Cannot connect to ldap server.");
      return PAM_AUTH_ERR;
   }

   char filter[255];
   char md5res[255];
   transfering_DATA((char*)hash);
   MD5_checksum((char*)hash,md5res);
    sprintf( filter ,"TUOCardMD5=%s",md5res);
   ret = lc_search(ld, NULL, filter, &result );
   if (ret != 0)
   {
      ret = lc_close(ld, result);
      pam_info(pamh, "Unknown user.");
      return PAM_USER_UNKNOWN;
   }
   char user[1024];
   user[0] = '\0';
    lc_ret_value( ld, result, "uid", 1, user );
   ret = lc_close(ld, result);
   pam_set_item(pamh,PAM_USER,user);

        return PAM_SUCCESS;
}
```

Výpis 7: Funkcia zabezpečujúca mapovanie

Modul najskôr získa dáta , ktoré program rfidlogin predal programu customlogin ako argument. Následne sa pripojí na LDAP server a vyhľadá užívateľa. Ak ho nájde, nastaví ho pomocou funkcie pam_set_item. Následne vráti správu o úspešnej autentifikácii.

7.6 Program rfidlogin

Tento program slúži na to, aby sledoval okrem štandardného vstupu aj vstup z čítacieho zariadenia. Na to, aby toto bolo možné, využíva konštrukciu jazyka c - *select*. Tento program sa vloží do prihlásovacieho procesu medzi program getty a program login.

Vzhľadom na to, že program rfidlogin je priamo spustený programom getty použitím funkcie *exec*, nemmáme možnosť programu predať parametre. Preto musíme na nastavenie programu použiť konfiguračný súbor. V tomto súbore nastavujeme tieto položky

- device - cesta k súboru, ktorý predstavuje čítacie zariadenie
- timeout - časový limit selectu

Obe tieto položky sú povinné. Konfiguračný súbor má názov *rfid.conf* a je umiestnený v adresári */etc/*. Tvar jedného riadku konfigurácie je *option=value;*.

```
device=/dev/ttyUSB0;
timeout=5;
```

Výpis 8: Výpis z konfiguračného súboru

7.6.1 Ako program funguje

Prvým krokom je náčítanie konfiguračného súboru. Funkcia , ktorá to zabezpečuje má 3 parametre.

- filename - smerník na reťazec znakov s absolutnou cestou konfiguračného súboru
- device - smerník na reťazec znakov, do ktorého sa uloží cesta k súboru čítacieho zariadenia
- timeout - referencia na premennú typu int, do ktorej sa uloží hodnota timeout

```
void loadConfig(char* filename , char* device, int& timeout)
{
    char line[30];

    FILE* file ;
     file  = fopen(filename,"r") ;
    if ( file )
    {
        while(fscanf( file , "%s;",line )  != EOF)
        {
```

```
            for(int i = 0; i< sizeof(line);i++)
            {
                if(line[i] == ';')
                    line[i] = '\0';
            }

            if (!strncmp("device=",line,7))
            {
                strcpy(device,line+7);
            }

            if (!strncmp("timeout=",line,8))
            {
                timeout = atoi(line+8);
            }
        }
        fclose(file);
    }
}
```

Výpis 9: Parsovanie konfiguračného súboru

Keď už máme konfiguráciu načítanú, môžeme zahájiť čítanie vstupov. Funkcia *select* slúži na synchrónne multiplexovanie vstupov alebo výstupov. V jenom z parametrov jej predáme množinu identifikátorov súborov (file descriptor). Do tejto množiny vložíme file descriptor štandardného vstupu spolu s file descriptorom, ktorý vznikne pri otvorení čítacieho zariadenia.

```
fd_set fdset;
FD_ZERO(&fdset);
FD_SET(STDIN_FILENO, &fdset);
int port;
port = open(conf_device, O_RDONLY);
if(port == -1)
{
    fprintf(stderr,"Message: RFID reader currently unavailable.\n");
}
else
{
    /* PORT OPTIONS */
    struct termios opt;
    tcgetattr(port, &opt);
    cfsetispeed(&opt, B9600);
    cfsetospeed(&opt, B9600);
    tcsetattr(port, TCSANOW, &opt);
    FD_SET(port,&fdset);
}
```

Výpis 10: Príprava množiny file descriptorov

Následne voláme funkciu select a predáme jej pripravenú množinu spolu s časovým limitom. Tým zabezpečíme multiplexovanie štandardného vstupu a vstupu z čítacieho zariadenia. Ostáva ešte definovať obsluhu k jednotlivým typom vstupu.

Vstup zo štandardného vstupu obslúžime tak, že prečítame hodnotu a následne pomocou funkcie *exec* spustíme program login. Programu predáme prečítanú hodnotu ako parameter. Prihlásenie v tomto prípade vlastne prebehne tak, ako za normálnych okolností. Rozdiel je len v tom, že užívateľské meno nenačítava program getty ale náš nový program.

Vstup z čítacieho zariadenia oblúžime podobným spôsobom. Rozdiel je v tom, že po načítaní dát z RFID tagu spustíme upravenú verziu programu login - customlogin s vlastným konfiguračným súborom v adresári */etc/pam.d/*. Okrem toho program customlogin dostane ako užívateľské meno dáta z RFID karty. Takže v tomto štádiu vlastne ešte nevieme užívateľské meno. O mapovanie na správneho užívateľa sa stará modul pam_ldap_mapping, ktorý na základe dát z karty vyhľadá v LDAPe správneho užívateľa a nasledne ho zmení.

Z tohoto dôvodu je modul označený kontrolným tagom REQUISITE. Ak je mapovanie užívateľa neúspešné, nemá zmysel ďalej pokračovať v autentifikácii nakoľko program customlogin ako užívateľské meno vždy dostáva dáta z RFID karty a takého užívateľa v systéme nemáme.

```
if ( FD_ISSET( STDIN_FILENO, &fdset ) )
{
   char buf[100] = "";
    int l = read( STDIN_FILENO, buf, sizeof( buf ) );
    if ( l < 0 )
       printf ("cannot read from stdin");
    if (strcmp(trim(buf),"" ) == 0)
      return 0;
    int argn=0;

    char* argv[3];
    argv[argn++] = "--";
    argv[argn++] = trim(buf);
    argv[argn++] = NULL;

    execv("/bin/login",argv);
}
```

Výpis 11: Obsluha pre štandardný vstup

```
if ( port != -1 && FD_ISSET( port, &fdset))
{
   char resp[255];
   char data[255];
    char rfiddata[255];
```

```
    int counter = 0;
    do
    {
        int l = read(port, resp, sizeof(resp));
       memcpy(data+counter, resp, l);
       counter += l;
    } while (counter < 11);

    close(port);
     if (data[0] == STX)
     {
       strncpy(rfiddata, data+1,10);
          rfiddata[10]='\0';
          int argn=0;
          char* argv[3];
          argv[argn++] = "--";
          argv[argn++] = rfiddata;
          argv[argn++] = NULL;
          execv("/bin/customlogin", argv);
      }
}
```

Výpis 12: Obsluha pre vstup z čítacieho zariadenia

7.7 Program customlogin

Tento program je vlastne program login so zmeneným názvom PAM služby. Tento program je spušťaný programom rfidlogin. Využíva sa pri prihláseni kartou v režime tty. Ako užívateľské meno dostava dáta z RFID karty. Toto meno je neskôr zmenené modulom pam_ldap_mapping. Modifikácia zdrojového kódu je znázornená nižsie 13.

```
retcode = pam_start ("customlogin", username, &conv, &pamh);
if (retcode != PAM_SUCCESS) {
    fprintf (stderr,
    _("login: PAM Failure, aborting: %s\n"),
    pam_strerror (pamh, retcode));
   SYSLOG ((LOG_ERR, "Couldn't initialize PAM: %s",
    pam_strerror (pamh, retcode)));
   exit (99);
}
```

Výpis 13: Ukážka modifikácie programu login

Názov PAM pam služby je nastavený na *customlogin*. Autentifikácia sa bude konfigurovat' v súbore */etc/pam.d/customlogin*.

8 Nasadenie

V tejto sekcii si zhrnieme, čo všetko je potrebné vykonať pre nasadenie systému. Uvedieme zoznam aplikácii, ktoré je potrebné do systému vložiť, príklady konfigurácie, zoznam balíčkov ktoré je potrebné nainštalovať.

8.1 Kroky potrebné pre nasadenie

- Do adresára /bin/ vložiť programy rfidlogin a customlogin.
- Do adresára /lib/security/ vložiť moduly pam_ldap_mapping.so a pam_rfid_mapping.so.
- Do adresára /etc/ vložiť konfiguračný súbor rfid.conf priložený k aplikácii rfidlogin.
- Do adresára /etc/pam.d/ vložiť súbor rfid-auth ktorý obsahuje pam konfiguráciu pre prihlásenie kartou. Tento súbor budeme vkládať do konfigurácie programu gdm direktívou @include.
- Upraviť konfiguráciu súboru /etc/inittab.
- Do systému vložiť užívateľa *rfid*.
- Upraviť konfiguráciu v /etc/pam.d/customlogin, /etc/pam.d/gdm3.
- Nastaviť Name Service Switch v súbore /etc/nsswitch.conf tak aby získaval užívateľské mena z LDAPu.
- V súbore /etc/gdm3/daemon.conf povoliť autologin a nastaviť na uživateľa rfid.

Poznámka 8.1 Program customlogin je modifikovaný program login z originálneho balíčka shadow. Na nainštalovanie do systému slúži script *install.sh*, ktorý je umiestnený v prílohe v adresári *shadow-4.1.4.2+svn3283*.

8.2 Konfigurácia PAM

- Konfigurácia PAM pre customlogin - spočíva vo vložení modulu pam_ldap_mapping do konfiguračného súboru. Tento modul označíme kontrolným flagom REQUISITE.
- Konfigurácia gdm - direktívou @include vložíme súbor rfid-auth pred moduly, ktoré zabezpečujú štandardný spôsob autentifikácie.

```
auth    sufficient    pam_rfid_mapping.so user=rfid device=/dev/ttyUSB0
auth    requisite     pam_succeed_if.so user != rfid
```

Výpis 14: Obsah súboru rfid-auth

Modulu pam_rfid_mapping nastavíme užívateľa, ktorého pokusom o prihlásenie sa spustí autentifikácia na základe dát z RFID karty. Modul následne zmení užívateľa na novo získaneho. Druhým parametrom je čítacie zariadenie.

Druhý riadok konfigurácie zabezpečí to, že ak sa užívateľ chce autentifikovať kartou (zadaním užívateľského mena definovaného v konfigurácii modulu) a z nejakého dôvodu je autentifikácia neúspešná, nedôjde k prihláseniu na užívateľa rfid. Neúspech môže nastať napríklad vtedy, keď s danou kartou v LDAPe nie je asociovaný žiadný užívateľ. Neúspech nastane aj vtedy, ak je nedostupný LDAP server alebo je odpojené čítacie zariadenie.

8.3 Výpis výsledných konfigurácii

8.3.1 Konfiguračný súbor /etc/inittab

```
1:2345:respawn:/sbin/getty -l /bin/ rfidlogin -n 38400 tty1
2:23:respawn:/sbin/getty -l /bin/ rfidlogin -n 38400 tty2
3:23:respawn:/sbin/getty -l /bin/ rfidlogin -n 38400 tty3
4:23:respawn:/sbin/getty -l /bin/ rfidlogin -n 38400 tty4
5:23:respawn:/sbin/getty -l /bin/ rfidlogin -n 38400 tty5
6:23:respawn:/sbin/getty -l /bin/ rfidlogin -n 38400 tty6
```

Výpis 15: Výpis z konfiguračného súboru /etc/inittab

8.3.2 Konfiguračný súbor /etc/pam.d/customlogin

```
auth        optional    pam_faildelay.so  delay=3000000
auth [success=ok new_authtok_reqd=ok ignore=ignore user_unknown=bad default=die]
     pam_securetty.so
auth        requisite   pam_nologin.so

auth      requisite   pam_ldap_mapping.so
session [success=ok ignore=ignore module_unknown=ignore default=bad] pam_selinux.so close
session       required    pam_env.so readenv=1
session       required    pam_env.so readenv=1 envfile=/etc/default/locale

auth        optional    pam_group.so
session     required    pam_limits.so
session     optional    pam_lastlog.so
session     optional    pam_motd.so motd=/run/motd.dynamic
session     optional    pam_motd.so

session     optional    pam_mail.so standard
@include common-account
@include common-session
@include common-password
```

Výpis 16: Výpis z konfiguračného súboru /etc/pam.d/customlogin

8.3.3 Konfiguračný súbor /etc/pam.d/gdm3

```
auth      requisite          pam_nologin.so
auth   required pam_succeed_if.so user != root quiet_success
auth    sufficient    pam_succeed_if.so user ingroup nopasswdlogin

@include rfid-auth
@include common-auth

auth      optional           pam_gnome_keyring.so
@include common-account

session [success=ok ignore=ignore module_unknown=ignore default=bad]    pam_selinux.so close
session required   pam_mkhomedir.so
session required          pam_limits.so
session required          pam_env.so readenv=1
session required          pam_env.so readenv=1 envfile=/etc/default/locale
session required          pam_loginuid.so
@include common-session

session [success=ok ignore=ignore module_unknown=ignore default=bad]    pam_selinux.so open
session optional          pam_gnome_keyring.so auto_start
@include common-password
```

Výpis 17: Výpis z konfiguračného súboru /etc/pam.d/gdm3

8.4 Zoznam závislostí

- libpam-modules, libpam0g : PAM moduly
- libpam0g-dev : Obsahuje hlavičkové súbory potrebné pre vývoj PAM modulov a aplikácii.
- libldap, libldap2-dev : Knižnice pre prístup do LDAPu.
- libssl, libssl-dev : Obsahuje knižnicu libcrypto s implementovaným šifrovacím algoritmom md5.

9 Záver

V úvode práce sme definovali cieľe práce použité technológie a prostriedky. Detailne som popísal technológiu RFID ako aj použíté čítacie zariadenie. Neskôr sme si povedali ako fungujú prihlásovacie a autentifikáčné mechanizmy v systéme linux, ako k dosiahnutiu cieľov využit PAM framework. Vyvinuli sme sériu aplikácii, ktoré nám umožnia prihlásenie na počítač modernejším spôsobom ako je zadávanie mena a hesla.

František Fiala

10 Reference

[1] Kenneth Geisshirt *Pluggable Authentification Modules* The Definitive Guide to PAM for Linux SysAdmins and C Developers

[2] Lukáš Jelínek *Vytváříme vlastní distribuci Linuxu* Od návrhu po fungující systém

[3] *RFID JOURNAL* www.rfidjournal.com

[4] *ID - 12 Innovations datasheet*

11 Prílohy

K práci je priložené DVD ktoré obsahuje:

- Adresár rfidlogin - zdorjové kódy programu rfidlogin + konfiguračný súbor rfid.conf
- Adresár pam_ldap_mapping - zdrojové kódy modulu pam_ldap_mapping.so
- Adresár pam_rfid_mapping - zdrojové kódy modulu pam_rfid_mapping.so
- Súbor rfid-auth - Konfiguračný súbor pre autentifikáciu pomocou RFID karty
- Adresár shadow-4.1.4.2+svn3283 - Zdrojové kódy balíčka shadow s upravenou verziou programu login